N-BEATS 是基于趋势性和季节性统计模型的深度学习算法，模型架构中后向和前向残差链接以及非常深的全连接层堆栈。框架设计的关键原则有：一、基础架构应该简单而通用，但也要足够深度地去挖掘潜在信息。二、架构不应依赖于输入数据特定于时间序列的特征工程或缩放。三、架构应该是可扩展的，可以被解释的。这些原则都有在 N-BEATS 的架构中得以体现。

Block

N-BEATS 模型中最小的堆栈单位为 block，1 个 block 有 4 个全连接层堆栈。如 Fig1 所示，N-BEATS 的特点在于采用了**双残差堆叠设计(Doubly Residual Stacking)**，后向(backcas)和前向(forecast)两个任务都应用了残差设计 。数据输入后，会有后向和前向两条运算路径，forecast 和 backcast，相邻全连接层通过残差连接。残差的设计意味着每个输出都被相减转换成残差，并作为模型更深部分的输入，这样下一个 block 的学习目标可专注于当前仍然无法解释的部分，保证了神经网络的深度。

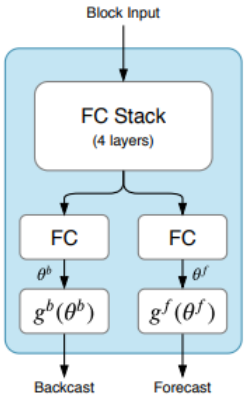

Figure 1 N-BEATS Block 框架

如 Fig2 所示，N-BEATS 设计了两个版本的内部结构：通用模型(generic)和可解释模型(trend&seasonality)。第一个用作黑匣子，而第二个将时间序列描述为趋势性和季节性的函数，即通过对 g^f 和 g^b 矩阵设置限制条件，训练过程本质上也就是求解 coefficients 和 basis vectors 的线性组合。

```python
num_samples, time_steps, input_dim, output_dim = 500, 10, 2, 1

# Definition of the model.
model_keras = NBeatsKeras(backcast_length=time_steps, forecast_length=output_dim,
                          stack_types=(NBeatsKeras.GENERIC_BLOCK, NBeatsKeras.GENERIC_BLOCK),
                          nb_blocks_per_stack=2, thetas_dim=(4, 4), share_weights_in_stack=True,
                          hidden_layer_units=64)
# Definition of the objective function and the optimizer.
model_keras.compile(loss='mae', optimizer='adam')

# Train the model.
print('Keras training...')
m=model_keras.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=256, batch_size=128)
```

```python
num_samples, time_steps, input_dim, output_dim = 500, 10, 2, 1

# Definition of the model.
model_keras = NBeatsKeras(backcast_length=time_steps, forecast_length=output_dim,
                          stack_types=(NBeatsKeras.TREND_BLOCK, NBeatsKeras.SEASONALITY_BLOCK),
                          nb_blocks_per_stack=2, thetas_dim=(4, 4), share_weights_in_stack=True,
                          hidden_layer_units=64)
# Definition of the objective function and the optimizer.
model_keras.compile(loss='mae', optimizer='adam')

# Train the model.
print('Keras training...')
m=model_keras.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=256, batch_size=128)
```

```python
def seasonality_model(thetas, backcast_length, forecast_length, is_forecast):
    p = thetas.get_shape().as_list()[-1]
    p1, p2 = (p // 2, p // 2) if p % 2 == 0 else (p // 2, p // 2 + 1)
    t = linear_space(backcast_length, forecast_length, is_forecast=is_forecast)
    s1 = K.stack([K.cos(2 * np.pi * i * t) for i in range(p1)])
    s2 = K.stack([K.sin(2 * np.pi * i * t) for i in range(p2)])
    if p == 1:
        s = s2
    else:
        s = K.concatenate([s1, s2], axis=0)
    s = K.cast(s, np.float32)
    return K.dot(thetas, s)


def trend_model(thetas, backcast_length, forecast_length, is_forecast):
    p = thetas.shape[-1]
    t = linear_space(backcast_length, forecast_length, is_forecast=is_forecast)
    t = K.transpose(K.stack([t ** i for i in range(p)]))
    t = K.cast(t, np.float32)
    return K.dot(thetas, K.transpose(t))
```

Figure 2 应用时在代码中调用的 block 堆栈设计（左）；源码中的季节性和周期性 block 的结构（右）

Stack

多个 block 进行堆叠就会组成一个 stack，block 之间通过残差连接。同时，**权重在同一个 stack 内共享**。

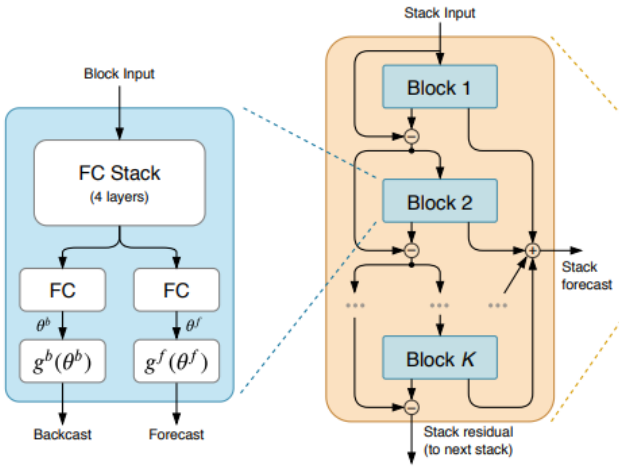

Figure 3 N-BEATS Stack 框架

如 Fig4 所示，Stack 之间也是通过残差进行堆叠，预测部分的残差会先在 Stack 内进行聚合，再在整个网络聚合。

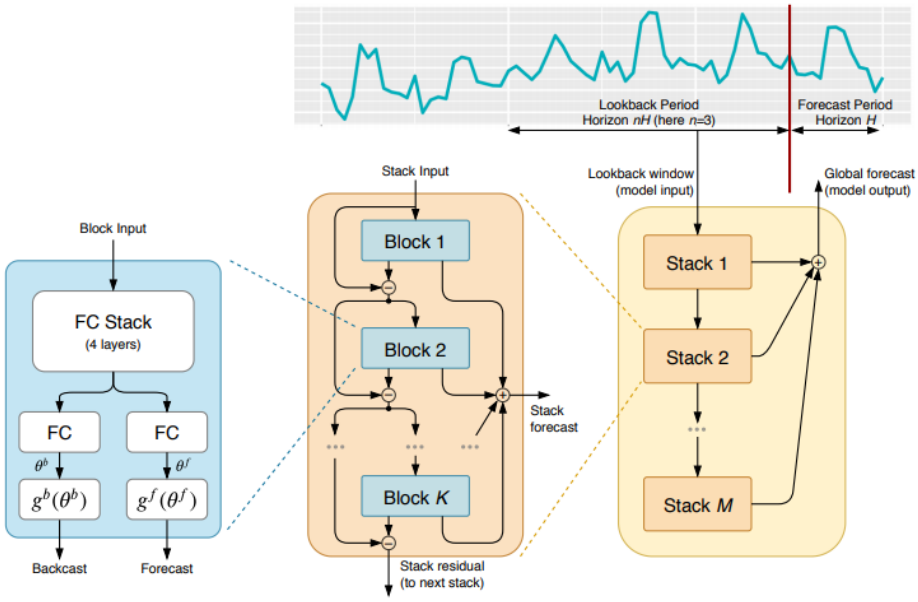

Figure 4 N-BEATS 完整结构

Figure 5 不同 stacking 设计的 N-BEATS 于单变量时间序列问题，季度性 GDP 数据集上的应用表现。（左）通用模型框架 generic；（右）可解释模型框架 trend&seasonality。

从 Fig5 可以观察到，通过不同 stack 设计的对比，通用框架（左）会比可解释框架（右）有更高的可释方差值，以及更好的拟合程度。在预测表现上，对 N_BEATS 引入可解释性的统计模型， 会需要权衡一定准确率，可释方差值降低 1.6%。

此外，经过在平滑性较高，时间间隔较大的年度 GDP 数据集上验证，通用框架和可解释框架的应用表现差别不大。这可被理解为，数据集本身波动性特点不大的前提下，调用 N_BEATS 可解释性模型不会对准确性带来影响。

对比其他算法在相同数据集上的表现，相比于 SVM，深度学习 N-BEATS 不依赖于特征工程和特征工程；相比于 LSTM，N-BEATS 展现了同时对于多变量和单变量时间序列数据集的强大适应性，保证了模型鲁棒性。

总体而言，N-BEATS 的准确性得益于其残差神经网络框架，每个传递和堆叠环节都由残差连接，能够挖掘足够深的潜在信息；N-BEATS 可解释性的优势在于在训练预测完成后，可参考限定统计模型的系数，为输出结果带来有意义的见解。对于 N-BEATS 设置可解释模型会权衡掉的准确率的问题，考虑到准确率仅降低 1.6%，在实际的经济预测应用中，N-BEATS 算法还有足够数据挖掘潜力与应用价值的。